**Санкт–Петербургский государственный университет**

***ЛАРИН Евгений Сергеевич***

## Выпускная квалификационная работа

# *Выделение именованных сущностей в текстах системы документооборота*

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2016 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Математическое и программное обеспечение вычислительных машин»

Научный руководитель:
доцент, кафедра технологии программирования,
к.т.н. Блеканов Иван Станиславович

Рецензент:
доцент, кафедра компьютерных технологий
и систем, к.ф. - м.н. Смирнов Михаил Николаевич

Санкт-Петербург
2020 г.

# Содержание

**Введение** . . . . . 3
Актуальность . . . . . 3
Цель работы . . . . . 4
Задачи работы . . . . . 4
Практическая значимость . . . . . 5
**Глава 1. Обзор существующих решений в области анализа текстов системы документооборота.** . . . . . 6
1.1. Обзор существующих готовых программных продуктов . 6
1.2. Обзор методов анализа текстов и выделения сущностей . 7
1.3. Обзор подходов оценки качества выделения сущностей . . 10
**Глава 2. Разработка программного комплекса для интеллектуального анализа текстов системы документооборота** . . . . . 12
2.1. Проектирование и реализация архитектуры программного комплекса . . . . . 12
2.2. Разработка методов интеллектуального анализа текстов системы документооборота . . . . . 16
2.3. Тестирование и оценка качества . . . . . 23
2.3.1 Постановка эксперимента . . . . . 23
2.3.2 Результаты . . . . . 25
2.3.3 Выводы . . . . . 29
**Заключение** . . . . . 30
Результаты работы . . . . . 30
Перспективы развития . . . . . 30
**Список литературы** . . . . . 31

# Введение

## Актуальность

В настоящее время задача анализа текста и, как частный случай, задача выделения именованных сущностей в текстах стоит особо остро. Появился запрос у общества [1] (косвенно описано в [2]) на виртуальных цифровых помощников (Apple Siri, Яндекс.Алиса)[3, 4] и различные интеллектуальные системы, работающие в узких предметных областях [5, 6, 7]. Помощники в узких областях призваны минимизировать человеческий фактор и вероятные ошибки в принимаемых решениях.

В условиях всё больше увеличивающихся масштабов документооборота вообще и электронного документооборота в частности, человеку становится всё труднее воспринимать и обрабатывать такое количество информации, что и порождает спрос на такие системы. Многие компании на рынке программных продуктов для бизнеса предлагают различные системы упрощающие и ускоряющие обработку информации.

Компания Digital Design [8] более 25 лет на рынке программного обеспечения, в том числе, в области решений «бизнес для бизнеса», поэтому ситуация, когда запрос общества и бизнеса на системы облегчающие (и ускоряющие) обработку текстов был бы проигнорирован, была невозможна.

Для успешного функционирования любых цифровых помощников необходимо техническое решение, позволяющие последнему «понимать» текст, а это означает, в частности, выделять именованные сущности: факты, события, действующих лиц и организации, нормативные акты и большое количество иной информации, легко воспринимаемой человеком, при этом, не всегда имеющей чёткую структуру.

Выделить именованную сущность означает — либо найти в тексте координаты (порядковые номера слов в тексте) её начала и конца, а также определить тип сущности, либо определить тип сущности и заполнить поля («свойства») соответствующие определенному на предыдущем шаге типу. В данной работе под выделением именованных сущностей подразумевается

именно поиск координат начала и конца именованной сущности.

Обработка естественного языка (NLP) зародилась ещё в 50-е годы прошлого столетия, хотя существуют работы в этой области, появившиеся ещё раньше. За это время эта область расширилась, нашла применение в реальной жизни и породила достаточно большое количество дочерних задач. Одна из них — выделение именованных сущностей.

Компанией Digital Design была поставлена задача создать программный комплекс, решающий задачу выделения именованных сущностей в текстах системы документооборота.

В данной работе рассмотрены готовые программные решения, которые могли бы быть применены к решению этой задачи, проанализированы различные методы и подходы к её решению, выбрана оптимальная конфигурация, проведена разметка данных и создан программный комплекс.

## Цель работы

Целью работы является разработка программного комплекса интеллектуального анализа текстов системы документооборота на основе нейросетей, который можно было бы использовать для создания системы всплывающих подсказок, дающих краткое описание документа, организации, частного лица или иной ссылки, встреченной в рассматриваемом документе. Эти два комплекса помогут решить проблему минимизации человеческого фактора в принимаемых решениях в различных юридических и законодательных процессах.

## Задачи работы

Для достижения цели были поставлены следующие задачи:

- Обзор существующих готовых программных продуктов в области выделения именованных сущностей и методов интеллектуального анализа текста (выделения именованных сущностей [9, 10, 11, 12], токенизация текста [13, 14] и др.)
- Обзор форматов разметки и утилит разметки текстовых коллекций для выделения именованных сущностей

- Выбор архитектуры программного комплекса
- Разработка программного комплекса
- Разработка скриптов, переводящих формат выбранной утилиты для разметки (brat [15]) в формат, воспринимаемый модулем предсказаний и обратно
- Создание возможности предварительно автоматически разметить новые документы, что поможет эксперту меньше тратить времени на редактирование разметки
- Разметка текстовой коллекции
- Проведение экспериментов

## Практическая значимость

Разработанное решение позволило бы облегчить работу государственных служащих за счёт отсутствия необходимости поиска информации по упомянутым в документе гражданам, организациям, органам власти и т.д., если его интегрировать в корпоративную систему документооборота и добавить краткие описания к найденным с помощью данного решения сущностям.

# Глава 1. Обзор существующих решений в области анализа текстов системы документооборота.

## 1.1 Обзор существующих готовых программных продуктов

На сегодняшний день на рынке представлено несколько решений (в том числе и имеющих открытый исходный код).

Одним из лучших решений является фреймворк DeepPavlov.ai [9, 10] с открытым исходным кодом [16], представленный Лабораторией нейронных систем и глубокого обучения МФТИ. Этот фреймворк имеет ряд плюсов: открытый исходный код и качественная документация даёт возможность использовать его практически в любых проектах связанных с выделением именованных сущностей. Этот фреймворк написан с использованием таких технологий как TensorFlow и Keras [16].

Также следует отметить тот факт, что команда разработчиков этого фреймворка вышла в полуфинал конкурса для вузов в области разговорного искусственного интеллекта от Amazon Alexa Prize Socialbot Grand Challenge 3 [17].

Из плюсов данного фреймворка стоит отметить тот факт, что это решение показывает одни из лучших результатов на русскоязычных коллекциях документов в задаче изменения именованных сущностей [18].

Второй большой плюс это реализация на кросс-платформенном языке программирования Python 3 [16] или, в качестве альтернативы, есть версия, упакованная в Docker-контейнер [19], имеющая интерфейс прикладного программирования (API), доступ к которому осуществляется по протоколу http [19]. Такие варианты реализации серьёзно упрощают развёртывание фреймворка на машине разработчика и клиента, а Docker-контейнер также предоставляет возможность написания интерфейса на любом удобном языке программирования.

Из особенностей можно отметить факт, что это не конечное решение, а библиотека, поэтому получение данных из pdf/A файлов необходимо производить на стороне клиента. Этот фреймворк защищён лицензией

Apache 2.0 [20].

Другим решением задачи выделения именованных сущностей можно считать Abbyy FlexiCapture, созданный российской компанией Abbyy, в котором также реализована система распознавания сущностей [11, 12]. Однако, в отличие от DeepPavlov.ai это, во-первых, готовый программный продукт, не требующий каких-либо доработок и модернизаций, во-вторых, универсальная платформа для интеллектуального анализа информации (т.е. здесь реализованы и методы получения текста из pdf/A [11, 21]), а в-третьих, эта платформа имеет закрытый исходный код, соответственно, её значительно сложнее адаптировать под специфическую задачу.

Из плюсов этого продукта стоит отметить достаточно богатый функционал: помимо выделения именованных сущностей этот комплекс ещё проводит классификацию документов, проверку документов по заданным правилам и справочникам, создаёт отчётность и может быть интегрирован в корпоративные информационные системы.

Интерфейс этой программы представлен в официальном блоге компании Abbyy [22] и выглядит как изображено на рисунках 1 и 2.

## 1.2 Обзор методов анализа текстов и выделения сущностей

На сегодняшний день появилось достаточно много методов выделения именованных сущностей, начиная от классических регулярных выражений (например, [23]) и заканчивая различными нейросетевыми технологиями (например, в [9, 10, 24, 25, 26, 27]). Из наиболее известных подходов к решению задачи выделения именованных сущностей стоит выделить разметку тегами последовательности (sequence tagging). Именно этот подход используется в методах, описанных далее. Основная идея этого подхода заключается в том, что получая на вход последовательность слов (как правило это фраза или предложение, то есть последовательность не особо длинная) метод возвращает последовательность меток, где каждому слову соответствует метка.

Есть два подхода в расставлении меток. Первый — предсказать рас-

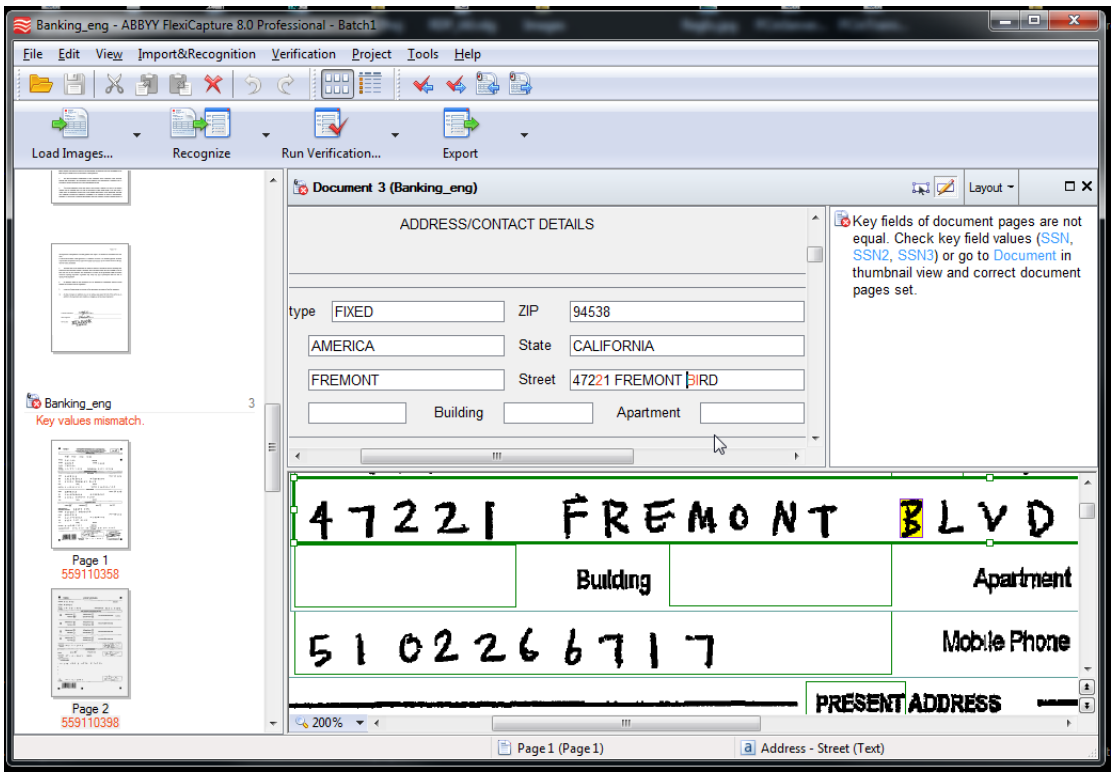



**Рис. 1:** Интерфейс Abbyy FlexiCapture, фото взято из [22].

пределение меток на каждом шаге, а затем применить лучеобразное декодирование для поиска лучшей последовательности меток (это классификатор максимальной энтропии и Марковские модели максимальной энтропии) [24]. Второй — перейти на уровень предложений с уровня слов. Здесь лидируют условные случайные поля [24]. У моделей CRF входы и выходы соединены напрямую (без использования обратных связей). Случайные условные поля [28], например, являлись state-of-the-art решением в 2013 году [24, 28, 29].

Также стоит учесть нейросетевые методы: Сети долгой краткосрочной памяти [24, 25, 26] и двунаправленные сети долгой краткосрочной памяти [24, 25, 26, 30], а также различные комбинации этих методов [24, 25, 26].

Было показано [24, 25, 26], что bi-LSTM превосходит LSTM, однако отстаёт от CRF, LSTM–CRF превосходит CRF в экспериментах, описанных в [24, 26], однако лучшие результаты показал bi-LSTM—CRF [24, 26]. Также более качественный результат bi-LSTM—CRF, по сравнению с bi-LSTM,

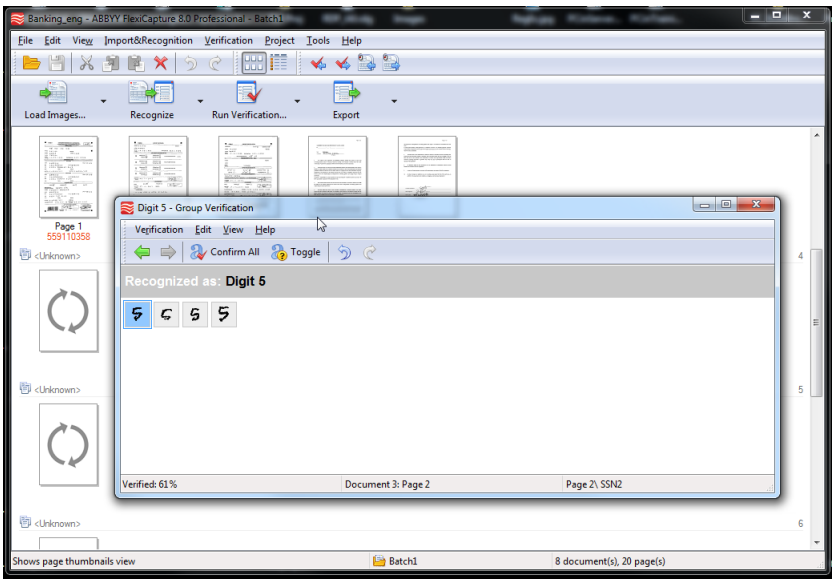



**Рис. 2:** Интерфейс Abbyy FlexiCapture, фото взято из [22].

показан в [25].

Однако стоит учесть, что на момент выхода [24, 25] ещё не было эмбеддинга BERT [14]. BERT — это двунаправленный энкодер, основанный на трансформерах. Согласно [14] BERT является контекстно зависимым энкодером, ставшим, к тому же лучшим решением для нескольких задач анализа текста [14].

Есть два подхода для применения BERT: использовать предобученную версию как эмбеддинг для других методов [26], или дополнительно тренировать его для решения определённой задачи [27]. Согласно [31], эмбеддинг играет архи-важную роль в эффективности разметки последовательности тегами. Это подтверждает работа [26]: применение BERT вместо W2V (который является контекстно независимым) существенно улучшило результат bi-LSTM—CRF.

Альтернативой подходу разметки тегами последовательности (sequence tagging) стоит отметить подход преобразования последовательности в последовательность (sequence2sequnce), использующийся в машинном переводе [32] и, даже, в задаче поиска кратчайшего пути между двумя вершинами графа [33]. В задаче выделения именованных сущностей этот подход также применяется [34].

## 1.3 Обзор подходов оценки качества выделения сущностей

Использовались оценки из библиотеки sklearn [35], аналогичные оценки, использовались во многих статьях, упомянутых в п. 1.2. Для каждой из категорий:

- precision [24, 25, 26, 28]:

$$precision = \frac{TP}{TP + FP}$$

- recall [24, 25, 26, 28]:

$$recall = \frac{TP}{TP + FN}$$

- f1-score [24, 25, 26, 28]:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Для результатов в целом:

- micro avg (рассчитывается TP, TN, FP и FN для всех категорий и считаются соответствующие оценки (т.е., например, TP — это сумма всех TP для всех категорий))
- macro avg (усреднённая по категориям соответствующая оценка)
- weighted avg (усреднённая, взвешенная по количеству представителей категорий, оценка)

В качестве вектора меток (предсказанных и истинных) используются соответственно предсказанные и истинные метки для каждого вектора части слова, сгенерированного эмбеддингом.

- TP (true positive) — количество меток, верно соотнесённых категории

- TN (true negative) — количество меток, верно не соотнесённых категории
- FP (false positive) — количество меток, неверно соотнесённых категории
- FN (false negative) — количество меток, неверно не соотнесённых категории

# Глава 2. Разработка программного комплекса для интеллектуального анализа текстов системы документооборота

## 2.1 Проектирование и реализация архитектуры программного комплекса

Решение было написано на языке программирования Python 3 [36]. Был выбран именно этот язык программирования, так как:

- Это кросс-платформенный язык, то есть программный комплекс не придётся адаптировать под каждую операционную систему;
- Для этого языка написано достаточно много библиотек, позволяющих относительно просто реализовывать различные методы интеллектуального анализа текста;
- Существует возможность достаточно просто интегрировать программный комплекс, написанный на этом языке, в веб-сервис;

За основу взята библиотека от sberbank-ai ner-bert [37], которая была адаптирована под рассматриваемые данные.

Для получения текстов из файлов формата pdf/A использовалась свободно распространяемая библиотека PDFMiner [38], которая защищена лицензией MIT [39]. Эта лицензия позволяет использовать данную библиотеку для любых целей.

После получения текста из pdf/A файла осуществляется очистка его от «мусорных»символов, а также удаление штампа документа. После предобработки, текст документа дробится по страницам или по выставленному параметру длины фрагмента (в зависимости от того, что наступит раньше). В идеале нужно дробить по предложениям, однако, учитывая специфику данных и большое количество мусорных символов, являющихся ошибкой распознавания текста в отсканированных документах, оказалось проще и эффективнее сделать разбиение по страницам и заданному параметру длины одного «предложения». Таким образом, все «предложения»имеют длину, меньше некоторого, наперёд заданного параметра. Затем данные сохра-

няются в виде двух файлов: текста и меток. Каждая метка соответствует одному и только одному слову.

Файл с метками может быть заполнен автоматически (предварительная разметка) посредством использования модели, здесь используемой, или может быть создан пустым и заполнен экспертом. Для разметки экспертом, а также для визуализации и исправления результатов автоматической разметки, используется утилита Brat Annotation Tool [15], которая обладает целым рядом преимуществ:

- Во-первых, это утилита, работающая на основе сервера Apache [40] и веб-браузера, что позволяет нескольким экспертам работать с одной коллекцией;
- Во-вторых, формат данных для этой утилиты наиболее близок к выбранному ранее формату данных (который, в свою очередь, близок формату данных для библиотеки [37]);
- В-третьих, эта утилита позволяет создать визуализацию разметки, что существенно упрощает (и ускоряет) работу;

Интерфейс этой утилиты представлен на рисунке 3.

Во время разметки выделяются только сущности. Их связи игнорируются.

Помимо Brat Annotation Tool есть скрипт, строящий представление для пользователя из формата, воспринимаемого модулем предсказаний. Это представление является текстовым файлом с выделенными в нём сущностями (сущность берётся в квадратные скобки, после последнего слова сущности ставится её название). Это представление бывает полезно для проверки результатов работы Brat Annotation Tool и скриптов перевода из её формата в формат, воспринимаемый модулем анализа текста. Эта проверка бывает очень полезна, так как эта утилита разметки допускает разметку частей слова разными тегами, что не допускается модулем анализа текста.

Были написаны скрипты перевода из одного формата в другой. Сама модель схожа по своей сути с моделью, представленной в [26], с разницей в размерах слоёв и в варианте предобученной модели BERT.

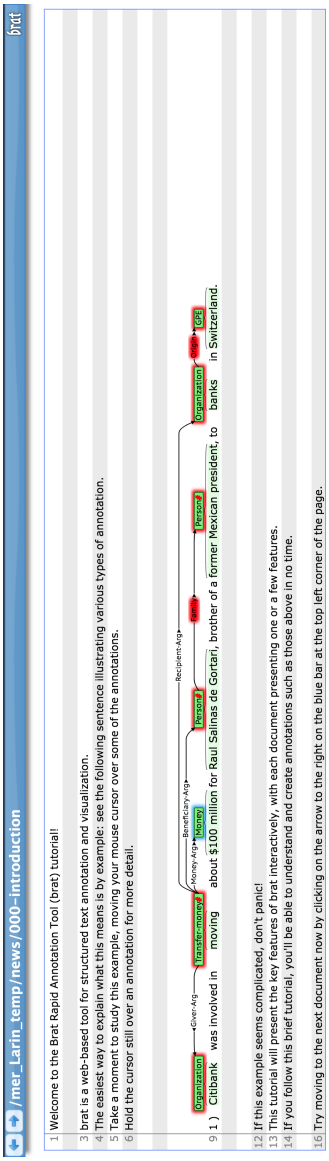



**Рис. 3:** Интерфейс Brat Annotation Tool.

Для оптимизации применяется алгоритм Adam, взятый из библиотеки pytorch-pretrained-BERT[41]. Количество эпох — 100, валидация на каждом шаге. Оптимизируется метрика macro F1 (см. п.1.3), т.к. классы не сбалансированы. Вычисление метрик осуществляется с помощью функции classification_report из библиотеки sklearn [35] с последующим получением из этого отчёта всех значений для вычисления статистических характеристик.

Валидация, обучение и тестирование проводятся на непересекающихся множествах. Обученной моделью считается такое её состояние, на кото-

ром macro F1 мера на валидационном множестве максимальна. При обучении множество предложений дробится на пакеты, которые обрабатываются по одному в период. Каждый пакет содержит несколько предложений, максимальная совокупная длина которых определяется соответствующим параметром. Сначала производится расчёт выхода bi-LSTM (для обоих направлений), получаем некоторые «очки» для каждой метки каждого слова, затем решается оптимизационная задача CRF (вычисляется матрица оценки перехода и, с её помощью, оптимальная последовательность тегов). Далее происходит вычисление градиента для выхода сети и матрицы оценки перехода, которая используется в CRF. Затем происходит оценка ошибки и обновление весов bi-LSTM.

Модель обучается с помощью некоторой модификацией 5-Fold Validation (K-Fold Validation описана, например, в [42]). Основное отличие от классического варианта 5-Fold Validation заключается в том, что множество данных дробится не на две части (обучающее и тестовое), а на три (обучающее, тестовое и валидационное), соотношение между которыми $\frac{3}{5}$, $\frac{1}{5}$ и $\frac{1}{5}$ соответственно. Наличие отдельного валидационного множества обеспечивает более качественное обучение (благодаря отсутствию пересечения между валидационным и обучающим множеством) и тестирование (благодаря отсутствию пересечения между валидационным и тестовым множеством).

Перед дроблением на 5 частей множество перемешивается с помощью функции sample из библиотеки для анализа данных в Python 3 Pandas [43]. Само дробление производится с помощью функции array_split, являющейся частью математической библиотеки в Python3 NumPy [44]. Использование этих библиотек может быть обосновано тем, что эти решения рассчитаны на обработку больших объёмов данных.

При обучении с помощью модификацией 5-Fold Validation создаётся 6 файлов весов: состояние модели до обучения для проведения кросс валидации. Это решение позволяет проводить оценку методом 5-Fold Validation некоторой предобученной модели, например, на новостях, текстах Википедии, открытых датасетах, и 5 вариантов весов, полученных при обучении на каждом варианте обучающего и валидационного множества. По

результатам проверки на тестовом множестве производится выбор лучшей модели по метрике macro F1 score. Реализована возможность изменения конфигурации и/или полной замены нейросетевой модели на какую-либо другую модель, которая просто должна иметь определённый набор функций и параметров, что может быть очень актуально, если будет создана новая архитектура, которая станет новым state-of-the-art решением.

В режиме авторазметки модель может обучиться (если установлен соответствующий параметр) без использования алгоритма 5-Fold Validation или использовать веса полученные ранее из 5-Fold Validation. После каждого шага алгоритма (получение текста, разбиения по «предложениям», авторазметка, приведение к формату Brat Annotation Tool) производится сохранение на жёсткий диск, что повышает стабильность работы комплекса.

Сохранение авторазмеченных файлов осуществляется в автоматически создаваемые директории со специальными названиями, что позволяет сохранить информацию об идентификаторах и названиях файлов и позволяет однозначно соотнести размеченный автоматически или вручную текст и исходный файл.

Реализованы режимы работы с интерфейсом Nvidia Cuda и без него, а также реализована поддержка операционных систем Microsoft Windows, Linux Ubuntu и Apple Mac OS.

Алгоритм работы комплекса можно представить в виде общей схемы, изображённой на рисунке 13

## 2.2 Разработка методов интеллектуального анализа текстов системы документооборота

В качестве эмбеддинга взята предобученная для русского языка модель BertModel из библиотеки pytorch-pretrained-BERT[41]. BERT[14] — это контекстно зависимый энкодер, кодирующий не одно слово, а учитывающий левый и правый контекст. Это энкодер, основанный на трансформерах. Согласно [14] и [35] BERT является контекстно зависимым энкодером, ставшим, к тому же лучшим решением для нескольких задач анализа

текста [14]. В данной работе используется только предобученная версия BERT, никакого дообучения не производилось. Для выделения сущностей используются суперпозиция двунаправленных сетей долгой краткосрочной памяти bi-LSTM (прямой слой учитывает левый контекст, обратный — правый, их выходы объединяются[24]) и Условных Случайных полей. Согласно [24, 26] именно такая конфигурация обеспечивает лучший результат для задачи выделения именованных сущностей. На момент выхода [24] (2015 год) BERT ещё не было (он появился в 2018 [14]). Именно этим объясняется отсутствие его упоминания в [24]. В [26] описана такая же конфигурация, как и в [24], однако в качестве энкодера уже используется BERT (причём именно конфигурация BERT—bi-LSTM—CRF показала лучшие результаты в исследовании [26]). Таким образом, конфигурация в данной работе тождественна [26] с точностью до размеров слоёв и версии BERT. Однако задача, решённая в [26] — это выделение именованных сущностей для китайского языка, но можно надеяться, что эмбеддинг сгладит различия между особенностями письменности и морфологического строя в русском и китайском языках.

Двунаправленная сеть долгой краткосрочной памяти (bi-LSTM) есть дальнейшее развитие рекуррентной нейросети (RNN) [45]. Рекуррентная нейросеть генерирует выход не только на основе входных данных, но и на основе данных исторических [24]. Самая простая её схема изображена на рисунке 4.

Здесь фрагмент рекуррентной нейросети $A$ принимая на вход $x_t$ возвращает значение $y_t$. Обратная связь позволяет на каждом последующем шаге (или предыдущем, если передавать входной вектор в обратном порядке) учитывать информацию с предыдущего (последующего) шага [46]. Этот фрагмент можно представить в виде обычной нейросети, если его «развернуть»по времени $t$ (см. рисунок 5).

Здесь индексы означают «момент времени»(номер в последовательности) соответствующего входа и выхода. [46]

Основной недостаток RNN в том, что она не очень хорошо справляется с длиннодиапазонными зависимостями (с зависимостями между словами, расположенными далеко друг от друга) [46]. LSTM является частным

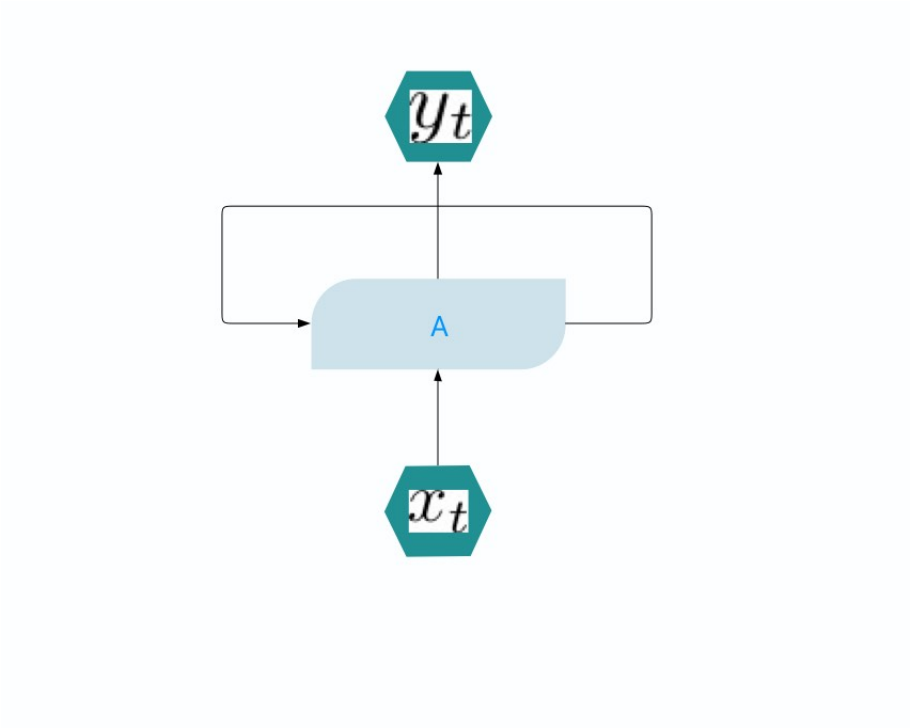



**Рис. 4:** Схема рекуррентной нейросети.

случаем RNN в котором, в отличие от классического, вместо скрытого слоя есть специальные ячейки памяти, которые обновляются не на каждом шаге (при появлении нового слова), а только тогда, когда приходит соответствующий сигнал из фильтра [24, 25, 30]. Поэтому они лучше подходят для поиска и использования длиннодиапазонных зависимостей.

На картинке 6 изображён фрагмент рекуррентной нейросети LSTM (на общих картинках рекуррентной нейросети обозначен символом $A$). Буквой $\sigma$ обозначена сигмоидная функция активации:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Символом $\times$ обозначено произведение Адамара [47] (покомпонентное произведение двух матриц). Значения переменных в синих кругах вычисляют-

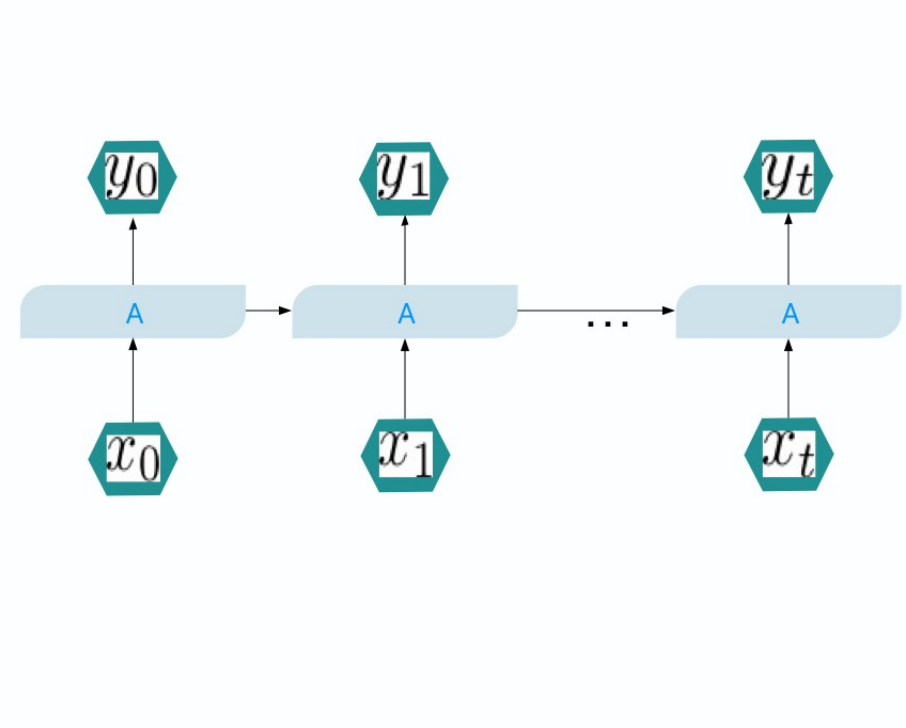



**Рис. 5:** «Развернутая»рекуррентная нейросеть (другое графическое представление).

ся по формулам [48]:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \times c_{t-1} + i_t \times g_t \\
h_t &= o_t \times tanh(c_t)
\end{aligned}
$$

- $i_t$ — элемент входа в момент $t$
- $f_t$ — элемент забывания в момент $t$
- $o_t$ — элемент выхода в момент $t$
- $g_t$ — элемент ячейки в момент $t$
- $c_t$ — состояние ячейки в момент $t$
- $h_t$ — скрытое состояние в момент $t$

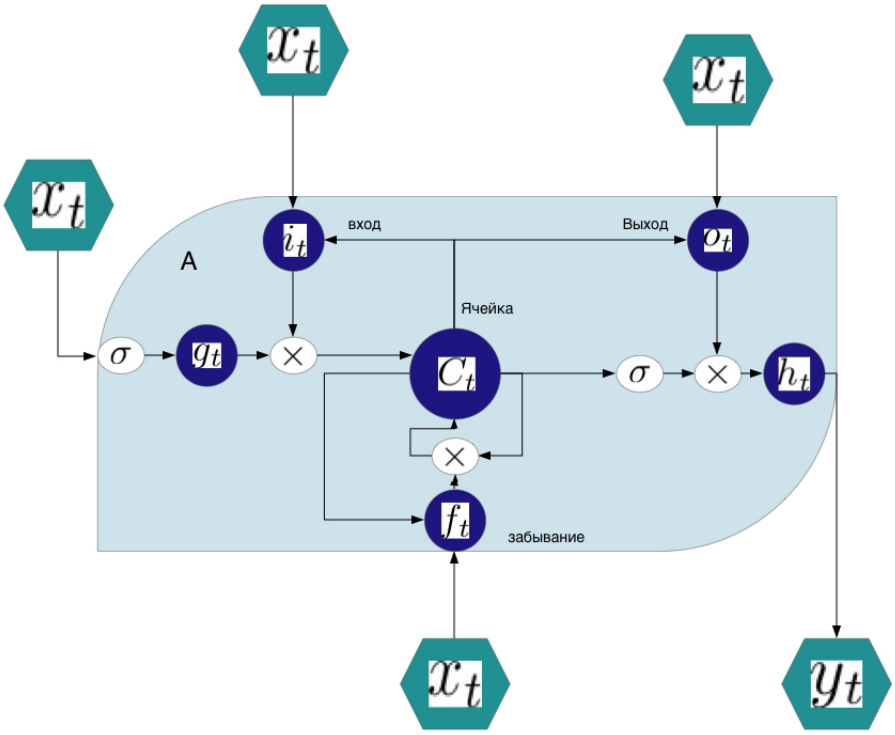



**Рис. 6:** Элемент сети долгой краткосрочной памяти.

Названия взяты из [48].

Особенностью данной разновидности RNN LSTM является то, что она воспринимает только левосторонний (или правосторонний, если вектор входа передать в обратном порядке) контекст. Так как обычно сущность в тексте зависит как от левостороннего, так и от правостороннего контекста целесообразнее (целесообразность доказана экспериментально, например в [24, 25]) использовать двунаправленную сеть долгой краткосрочной памяти (bi-LSTM). Этот вариант рекурсивной нейросети представляет собой комбинацию двух нейросетей долгой краткосрочной памяти, причём в одну из них входные данные передаются в обратном порядке. Это позволяет учитывать как левосторонний, так и правосторонний контекст. Схематично это можно представить как на рисунке 7.

Здесь слой A называется прямой, а B — обратный [24]. Параметры bi-LSTM:

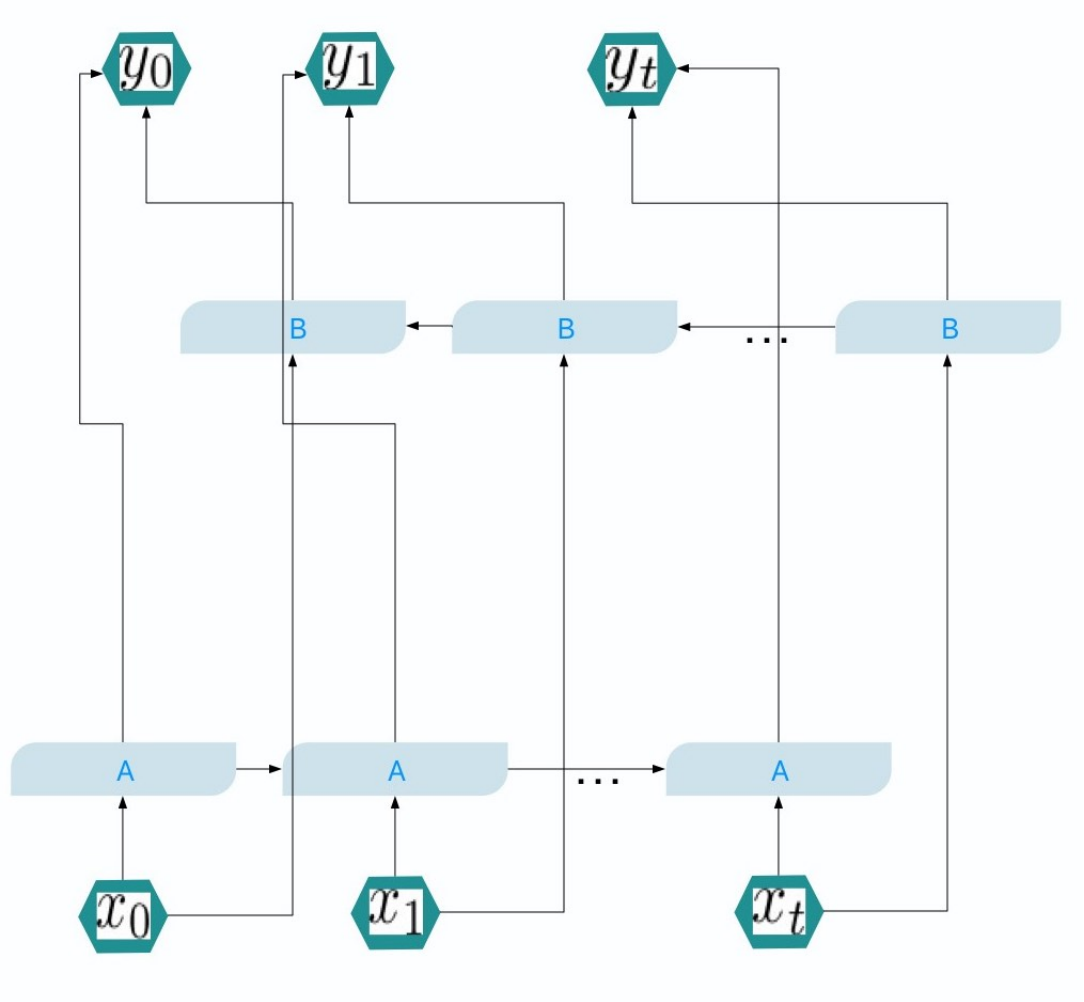



**Рис. 7:** Схема двунаправленной сети долгой краткосрочной памяти.

- Размерность входа: 768
- Размерность выхода: 256
- 1 слой rnn
- Dropout: 0

Параметры полносвязной линейной нейросети (она выполняет функцию линейного преобразования выхода bi-LSTM к размеру входа CRF $y = xA^T + b$ [48]):

- Входной слой: 512
- Выходной слой: количество меток + 5 служебных (метка слова, не относящегося ни к одной из категорий, фрагмент слова, не являющийся началом и т.д.)

- Dropout: 0.3

В качестве механизма расстановки меток используются Условные случайные поля (CRF). Условные случайные поля (CRF) — это ненаправленная дискриминативная графическая модель, являющаяся разновидностью Марковских случайных полей [28]. Тот факт, что это дискриминативная модель означает, что известна (вычислена эмпирически) вероятность $p(y|x)$, где $y$ — это последовательность меток, а $x$ — последовательность слов. Имея эту информацию можно классифицировать (найти вероятное значение скрытой переменной) по наблюдаемой величине [49]. У моделей CRF входы и выходы соединены напрямую (без использования обратных связей), чем она существенно отличается от рекуррентных сетей. Анализ производится на уровне всего предложения, а не на уровне отдельных слов [24].

Модель CRF имеет в качестве параметра матрицу оценки перехода, которая вычисляется при обучении. Оценка соответствия последовательности тегов предложению производится по следующей формуле:

$$s([x]_1^T, [y]_1^T, \widetilde{\theta}) = \sum_{t=1}^{T} (A_{[y]_{t-1},[y]_t} + [f_\theta]_{[y]_t t})$$

Здесь используются следующие обозначения:

- $[x]_1^T$ — последовательность слов (предложение, поданное на вход)
- $[y]_1^T$ — последовательность тегов
- $\widetilde{\theta}$ — настраиваемые параметры, $\widetilde{\theta} = \theta \cup \{A_{i,j} \forall i, j\}$, где $\theta$ — это параметры нейросети, а $[A]_{i,j}$ — матрица оценки перехода
- $[f_\theta]_{i,t}$ — это элемент матрицы выхода нейросети с параметрами $\theta$ для предложения $[x]_1^T$ для $i$-того тега и $t$-того слова

Задачу вычисления $[A]_{i,j}$ и оптимальной расстановки тегов можно решить с помощью динамического программирования [50, 51]. Это изложено в [52].

Параметры Условных случайных полей (CRF):

- количество меток + 5 + 2

Схему всего модуля интеллектуального анализа текстов можно представить как на рисунке 8, где буквами $x$ с индексами обозначены слова, подаваемые на вход, а $y$ — метки на выходе.

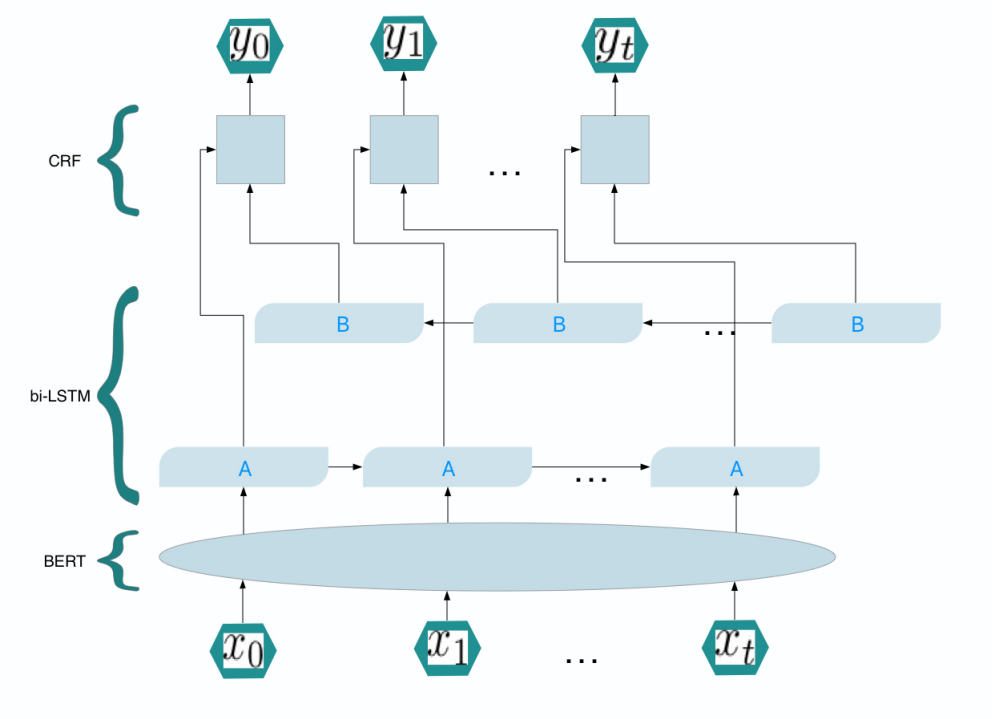



**Рис. 8:** Модуль интеллектуального анализа текста.

Использование такой архитектуры объясняется хорошими результатами, показанными этой архитектурой на аналогичных задачах в работах [24, 26], а также в проекте [37].

## 2.3 Тестирование и оценка качества

### 2.3.1 Постановка эксперимента

Была взята тестовая коллекция текстов системы документооборота на русском языке, размеченная автором, в которой были выделены следующие сущности:

- Федеральный орган власти
- Региональный орган власти

- Депутат
- Юридическое лицо (контрагент)
- Физическое лицо (Гражданин)
- Номер и название законопроекта
- Номер и название внешнего нормативного документа (федеральный закон, постановление, приказ и пр.)
- Номер и дата документа переписки

На вход программы подавался текст в формате pdf/A, его необходимо выделить, очистить от спец- и просто мусорных символов, выделить вышеуказанные сущности. Выходом программы является текстовый файл с отмеченными в нём сущностями.

Входными данными являются документы служебной переписки, отзывы на законопроекты и иные официальные документы, сохранённые в формате pdf/A.

Штамп не анализируется. Документы имеют достаточно большой разброс по длине. После предобработки (удаления мусорных символов, непечатных символов и знаков препинания) документ сохраняется в виде двух файлов одинаковой длины: текст и метки к каждому слову. Например, для входа *Росреестр издал приказ о приватизации в соответсвтии с ФЗ-000* выход будет выглядеть так: *росреестр издал приказ о приватизации в соответсвтии с ФЗ-000 от 00 декабря 0000 года* и *S_ Fed-gov O O O O O O O O O O O Outter-act I_ Outter-act I_ Outter-act I_ Outter-act I_ Outter-act L_ Outter-act*

В данный момент есть коллекция размеченных документов со следующими параметрами:

- Длина (записей): 125
- Всего слов: 12980
- Самая длинная запись (символов): 1603
- Самая короткая запись (символов): 1

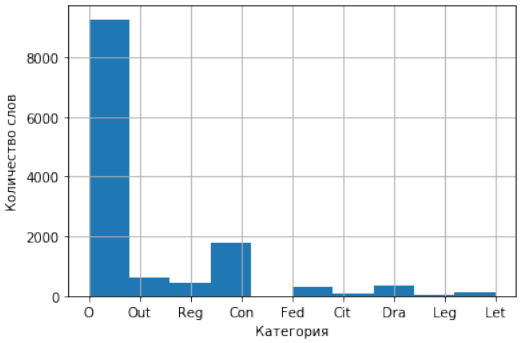



**Рис. 9:** Распределение слов по категориям. На графике используются следующие обозначения: «Out» — это номер и название нормативного документа, «Con» — это Юридическое лицо (контрагент), «Fed» — это Федеральный орган власти, «Reg» — это Региональный орган власти, «Dra» — это Номер и название законопроекта, «Leg» — это Депутат, «Let» — это Номер и дата документа переписки, «Cit» — это Физическое лицо (Гражданин), а «O» — это слова, не относящиеся к искомым категориям.

- Распределение слов по категориям: рисунок 9

Проверка качества производилась методом K-Fold Validation. Требовалось сравнить представленный здесь программный комплекс с имеющимися решениями по метрикам из пункта 1.3

### 2.3.2 Результаты

По результатам кросс-валидации можно построить две таблицы оценок: для валидационного множества и для тестового. В качестве значений для валидационного множества выбраны значения на таком шаге, на котором целевая метрика (macro F1 score) максимальна. Для каждого поля значение без скобок — это математическое ожидание, значение же в скобках — это дисперсия.

**Таблица 1:** Валидационное множество

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **I_Fed-Gov** | ***0.7314 (0.0207)*** | ***0.813 (0.0048)*** | ***0.7622 (0.0085)*** | ***64.8 (49.36)*** |
| **I_Reg-Gov** | ***0.751 (0.0175)*** | ***0.698 (0.0469)*** | ***0.7128 (0.0271)*** | ***86.6 (4207.04)*** |
| I_Letter-num-date | 0.5432 (0.1041) | 0.4154 (0.0829) | 0.4518 (0.0713) | 27.2 (566.96) |
| I_Citizen | ***0.7258 (0.1033)*** | ***0.684 (0.1109)*** | ***0.6984 (0.1053)*** | 16.2 (21.76) |
| I_Legislator | 0.0 (0.0) | 0.0 (0.0) | 0.0 (0.0) | 0.0 (0.0) |
| **I_Outter-act** | ***0.8 (0.0216)*** | ***0.7484 (0.0199)*** | ***0.7574 (0.0115)*** | ***119.0 (578.8)*** |
| **I_O** | ***0.9456 (0.0011)*** | ***0.973 (0.0002)*** | ***0.9586 (0.0002)*** | ***1845.0 (27141.2)*** |
| I_Draft-law | 0.6536 (0.1183) | 0.39 (0.0826) | 0.4472 (0.0801) | ***70.4 (2318.64)*** |
| **I_Contractor** | ***0.9372 (0.0024)*** | ***0.9416 (0.0004)*** | ***0.938 (0.0005)*** | ***351.0 (23257.2)*** |
| micro avg | 0.9236 (0.0004) | 0.9212 (0.0004) | 0.9224 (0.0004) | 2580.2 (62496.96) |
| **macro avg** | *0.707 (0.0012)* | *0.6584 (0.0001)* | ***0.6654 (0.0002)*** | 2580.2 (62496.96) |
| weighted avg | 0.9252 (0.0003) | 0.9212 (0.0004) | 0.9172 (0.0005) | 2580.2 (62496.96) |

Здесь используются следующие обозначения:

- I_O: Не относится к искомым категориям;
- I_Outter-act: Номер и название нормативного документа (федеральный закон, постановление, приказ и пр.);

- I_Legislator: Депутат;
- I_Fed-Gov: Федеральный орган власти;
- I_Reg-Gov: Региональный орган власти;
- I_Citizen: Физическое лицо (Гражданин);
- I_Letter-num-date: Номер и дата документа переписки;
- I_Draft-law: Номер и название законопроекта;
- I_Contractor: Юридическое лицо (контрагент);

В графе support - количество вхождений той или иной категории в рассматриваемом множестве.

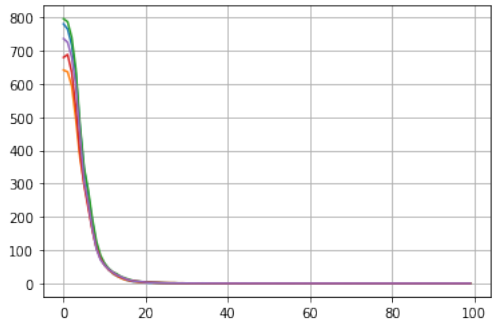



**Рис. 10:** Средняя арифметическая функции ошибки на каждой эпохе.

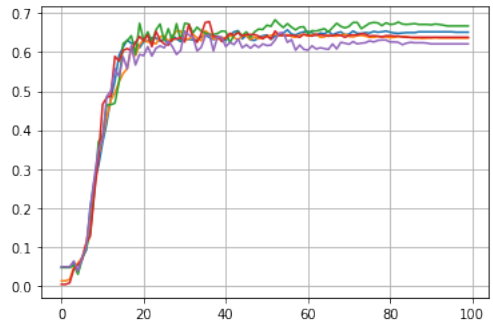



**Рис. 11:** Macro f1 меры для валидационного множества.

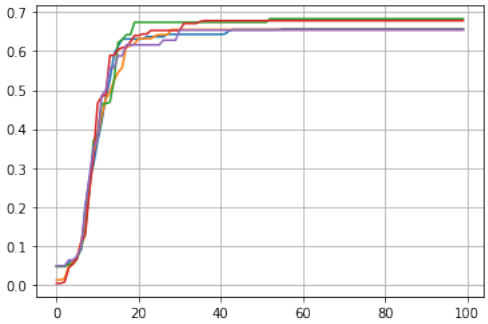



**Рис. 12:** Максимум macro f1 меры для валидационного множества.

**Таблица 2:** Тестовое множество

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **I_Fed-Gov** | ***0.7535 (0.0195)*** | ***0.7052 (0.0126)*** | ***0.7158 (0.0084)*** | ***64.8 (49.36)*** |
| **I_Reg-Gov** | ***0.7423 (0.0186)*** | ***0.7247 (0.0034)*** | ***0.7258 (0.0055)*** | ***86.6 (4207.04)*** |
| I_Letter-num-date | 0.5433 (0.0891) | 0.477 (0.1376) | 0.48 (0.109) | 27.2 (566.96) |
| I_Citizen | ***0.7269 (0.1375)*** | ***0.7083 (0.1403)*** | ***0.7131 (0.1347)*** | 16.2 (21.76) |
| I_Legislator | 0.0 (0.0) | 0.0 (0.0) | 0.0 (0.0) | 4.6 (84.64) |
| **I_Outter-act** | ***0.7872 (0.0406)*** | ***0.7476 (0.0072)*** | ***0.7491 (0.0121)*** | ***119.0 (578.8)*** |
| I_Draft-law | 0.5175 (0.1888) | 0.3082 (0.0665) | 0.3862 (0.1046) | ***70.4 (2318.64)*** |
| **I_Contractor** | ***0.9332 (0.0038)*** | ***0.9378 (0.0004)*** | ***0.9341 (0.0009)*** | ***351.0 (23257.2)*** |
| micro avg | 0.8544 (0.0047) | 0.7715 (0.0034) | 0.8097 (0.0032) | 739.8 (34223.76) |
| **macro avg** | *0.6255 (0.0069)* | *0.5761 (0.0046)* | ***0.588 (0.006)*** | 739.8 (34223.76) |
| weighted avg | 0.8468 (0.0046) | 0.7715 (0.0034) | 0.7965 (0.0038) | 739.8 (34223.76) |

В обеих таблицах жирным курсивом выделены те значения метрик, выборочное математическое ожидание которых выше макро усреднения соответствующей метрики и такие значения количеств вхождения (графа support), выборочное математическое ожидание которых больше 60. Жирным шрифтом выделены категории, показавшие приемлемые математические ожидания на достаточно большой выборке. Подчеркнутым курсивом выделено макро усреднение всех метрик, а macro f1-score к тому же выделена жирным шрифтом, так как в данной работе это целевая метрика.

Тот факт, что алгоритм сходится может быть подтверждён графиками функции ошибки (рисунок 10), macro f1 меры для валидационного множества (рисунок 11), а также максимума macro f1 меры для валидационного множества, на котором изображено максимальное значения на каждой эпохе в сравнении со всеми предыдущими эпохами (рисунок 12).

### 2.3.3 Выводы

По таблицам 1 и 2 отчётливо видно, что мало того, что слов не относящихся к искомым категорий значительно больше, чем слов, которые относятся к интересующим категориям, так ещё и сами категории имеют совершенно разный объём. Именно этим объясняются достаточно низкие результаты на тестовом множестве. Глядя на эти таблицы можно предположить, что для необходимых результатов будет достаточно $\sim 150$ вхождений каждой категории.

# Заключение

## Результаты работы

В ходе выполнения работы были выполнены следующие шаги:

- Проведён обзор существующих готовых программных продуктов в области выделения именованных сущностей и методов интеллектуального анализа текста
- Проведён обзор форматов разметки и утилит для разметки текстовых коллекций для выделения именованных сущностей
- Была выбрана архитектура программного комплекса
- Произведена разработка программного комплекса
- Произведена разработка скриптов, переводящих формат выбранной утилиты для разметки (brat [15]) в формат, воспринимаемый модулем предсказаний и обратно
- Была создана возможность предварительно автоматически разметить новые документы, что поможет экспертам меньше тратить времени на редактирование разметки
- Были проведены эксперименты

## Перспективы развития

Для того, чтобы улучшить результат, можно попытаться использовать RoBERTa вместо BERT, так как в [53] на похожей задаче RoBERTa показала более высокие результаты. Также использование более сбалансированного множества скорее всего приведёт к более точным (и более устойчивым) результатам. К тому же увеличение размеченного множества документов может существенно увеличить качество результатов.

# Список литературы


[1] Virtual Assistants: What They Do & How to Hire (2020 Update) // Time Doctor URL: https://biz30.timedoctor.com/what-does-a-virtual-assistant-do/ (дата обращения: 23.04.2020).

[2] Julia Kiseleva, Kyle Williams, Ahmed Hassan Awadallah, Aidan C. Crook, Imed Zitouni, and Tasos Anastasakos. 2016. Predicting User Satisfaction with Intelligent Assistants. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16). Association for Computing Machinery, New York, NY, USA, 45—54. DOI:https://proxy.library.spbu.ru:2060/10.1145/2911451.2911521

[3] V. Këpuska and G. Bohouta, «Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home)», 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, 2018, pp. 99–103.

[4] Патент: RU2654789C2: System and method for handling a spoken user request

[5] Goh K. J., Po S. B. Fertilizer recommendation systems for oil palm: estimating the fertilizer rates //Proceedings of MOSTA Best practices workshops-agronomy and crop management. Malaysian Oil Scientists' and Technologists' Association. – 2005. – С. 1–37.

[6] Eftimov T., Seljak B. K., Korošec P. A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations //PloS one. – 2017. – Т. 12. – №. 6.

[7] C. Sage, A. Aussem, H. Elghazel, V. Eglin and J. Espinas, «Recurrent Neural Network Approach for Table Field Extraction in Business Documents», 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, Australia, 2019, С. 1308–1313.

[8] О компании // Digital Design URL: https://digdes.ru/about (дата обращения: 15.05.2020).

[9] Sattarov O. Natural Language Processing with DeepPavlov Library and Additional Semantic Features //Artificial Intelligence. – Springer, Cham, 2019. – С. 146–159.

[10] An open source conversational AI framework URL: deeppavlov.ai (дата обращения: 19.05.2020).

[11] Возможности ABBYY FlexiCapture // Abbyy URL: https://www.abbyy.com/ru-ru/flexicapture/features/ (дата обращения: 19.05.2020).

[12] Holt X., Chisholm A. Extracting structured data from invoices //Proceedings of the Australasian Language Technology Association Workshop 2018. – 2018. – С. 53–59.

[13] Radford A. et al. Improving language understanding by generative pre-training //URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/languageunderstandingpaper.pdf. – 2018.

[14] Devlin J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding //arXiv preprint arXiv:1810.04805. – 2018.

[15] Stenetorp P. et al. BRAT: a web-based tool for NLP-assisted text annotation //Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics. – Association for Computational Linguistics, 2012. – С. 102–107.

[16] deepmipt/DeepPavlov // GitHub URL: https://github.com/deepmipt/DeepPavlov (дата обращения: 19.05.2020).

[17] Команда DREAM из МФТИ прошла в полуфинал конкурса по разработке искусственного интеллекта от Amazon МФТИ // МФТИ URL: https://mipt.ru/news/komanda_dream_iz_mfti_proshla_v_

polufinal_konkursa_po_razrabotke_iskusstvennogo_intellekta_ot_amazon (дата обращения: 19.05.2020).

[18] Features // DeepPavlov 0.9.1 documentation URL: http://docs.deeppavlov.ai/en/master/features/overview.html#ner-model-docs (дата обращения: 16.05.2020).

[19] deeppavlov's Profile // Docker Hub URL: https://hub.docker.com/u/deeppavlov (дата обращения: 16.05.2020).

[20] Apache 2.0 // GitHub URL: https://github.com/deepmipt/DeepPavlov/blob/master/LICENSE (дата обращения: 19.05.2020).

[21] Vlada M., Babiy I., Ivanescu O. ABBYY recognition technologies—-ideal alternative to manual data entry. Automating processing of exam tests //Star. – 2010. – Т. 3. – №. 1. – С. 3–8.

[22] Блог компании Abbyy // Хабр URL: https://habr.com/ru/company/abbyy/blog/ (дата обращения: 16.05.2020).

[23] Garofalakis M. N., Rastogi R., Shim K. SPIRIT: Sequential pattern mining with regular expression constraints //VLDB. – 1999. – Т. 99. – С. 7–10.

[24] Huang Z., Xu W., Yu K. Bidirectional LSTM-CRF models for sequence tagging //arXiv preprint arXiv:1508.01991. – 2015.

[25] Arkhipov M. Y. et al. Application of a Hybrid Bi-LSTM–CRF model to the task of Russian Named Entity Recognition //Conference on Artificial Intelligence and Natural Language. – Springer, Cham, 2017. – С. 91–103.

[26] Z. Dai, X. Wang, P. Ni, Y. Li, G. Li and X. Bai, "Named Entity Recognition Using BERT BiLSTM CRF for Chinese Electronic Health Records,"2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Suzhou, China, 2019, pp. 1–5.

[27] Labusch K., Neudecker C., Zellhöfer D. BERT for Named Entity Recognition in Contemporary and Historical German // Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019). 2019.

[28] Konkol M., Konopík M. CRF-based Czech named entity recognizer and consolidation of Czech NER research //International conference on text, speech and dialogue. – Springer, Berlin, Heidelberg, 2013. – C. 153–160.

[29] Lin, D., Wu, X.: Phrase clustering for discriminative learning. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, ACL 2009, vol. 2, pp. 1030-–1038. Association for Computational Linguistics, Stroudsburg (2009)

[30] Limsopatham N., Collier N. Bidirectional LSTM for named entity recognition in Twitter messages. – 2016.

[31] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research (JMLR) - 2011.

[32] Cho K. et al. On the properties of neural machine translation: Encoder-decoder approaches //arXiv preprint arXiv:1409.1259. – 2014.

[33] Bay A., Sengupta B. Sequence stacking using dual encoder Seq2Seq recurrent networks. – 2017.

[34] Straková J., Straka M., Hajič J. Neural architectures for nested NER through linearization //arXiv preprint arXiv:1908.06926. – 2019.

[35] Scikit Learn URL: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html` (дата обращения: 20.04.2020).

[36] Python 3.7.7 documentation URL: `https://docs.python.org/3.7/` (дата обращения: 20.04.2020).

[37] BERT-NER (nert-bert) with google bert // GitHub URL: `https://github.com/sberbank-ai/ner-bert` (дата обращения: 23.04.2020).

[38] PDFMiner documentation URL: `https://pdfminer-docs.readthedocs.io/pdfminer_index.html#` (дата обращения: 20.04.2020).

[39] Terms and Conditions // PDFMiner documentation URL: https://pdfminer-docs.readthedocs.io/pdfminer_index.html#terms-and-conditions (дата обращения: 20.04.2020).

[40] The Apache HTTP Server Project URL: https://httpd.apache.org/ (дата обращения: 20.04.2020).

[41] pytorch-pretrained-bert 0.6.2 // PyPI · The Python Package Index URL: https://pypi.org/project/pytorch-pretrained-bert/ (дата обращения: 22.04.2020).

[42] J. D. Rodriguez, A. Perez and J. A. Lozano, «Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation», in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 3, pp. 569–575, March 2010.

[43] pandas.DataFrame.sample // pandas 1.0.3 documentation URL: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sample.html (дата обращения: 20.04.2020).

[44] numpy.array_split // NumPy v1.17 Manual URL: https://docs.scipy.org/doc/numpy/reference/generated/numpy.array_split.html (дата обращения: 20.04.2020).

[45] Hammerton J. Named entity recognition with long short-term memory //Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4. – Association for Computational Linguistics, 2003. – С. 172–175.

[46] LSTM —- сети долгой краткосрочной памяти // Хабр URL: https://habr.com/ru/company/wunderfund/blog/331310/ (дата обращения: 25.04.2020).

[47] Horn R. A. The hadamard product //Proc. Symp. Appl. Math. – 1990. – Т. 40. – С. 87–169.

[48] PYTORCH DOCUMENTATION URL: https://pytorch.org/docs (дата обращения: 23.04.2020).

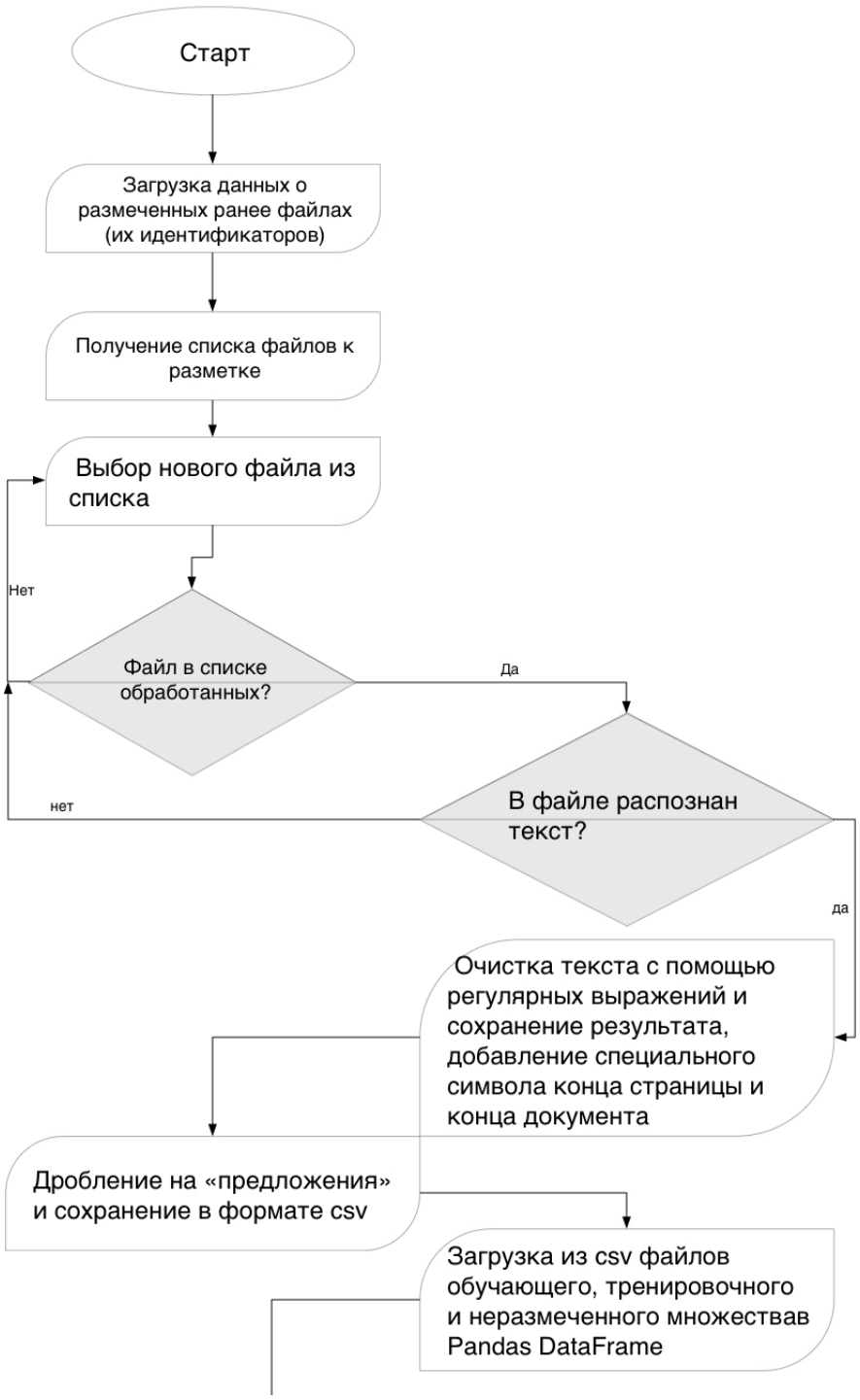
[49] Ng A. Y., Jordan M. I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes //Advances in neural information processing systems. – 2002. – С. 841–848.

[50] Rabiner L. R. A tutorial on hidden Markov models and selected applications in speech recognition //Proceedings of the IEEE. – 1989. – Т. 77. – №. 2. – С. 257–286.

[51] Viterbi A. J. Viterbi Algorithm //Wiley Encyclopedia of Telecommunications. – 2003.

[52] Lafferty J., McCallum A., Pereira F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. – 2001.

[53] Wang Y. et al. Application of Pre-training Models in Named Entity Recognition //arXiv preprint arXiv:2002.08902. – 2020.


Старт
Загрузка данных о размеченных ранее файлах (их идентификаторов)
Получение списка файлов к разметке
Выбор нового файла из списка
Нет
Файл в списке обработанных?
Да
нет
В файле распознан текст?
да
Очистка текста с помощью регулярных выражений и сохранение результата, добавление специального символа конца страницы и конца документа
Дробление на «предложения» и сохранение в формате csv
Загрузка из csv файлов обучающего, тренировочного и неразмеченного множествав Pandas DataFrame

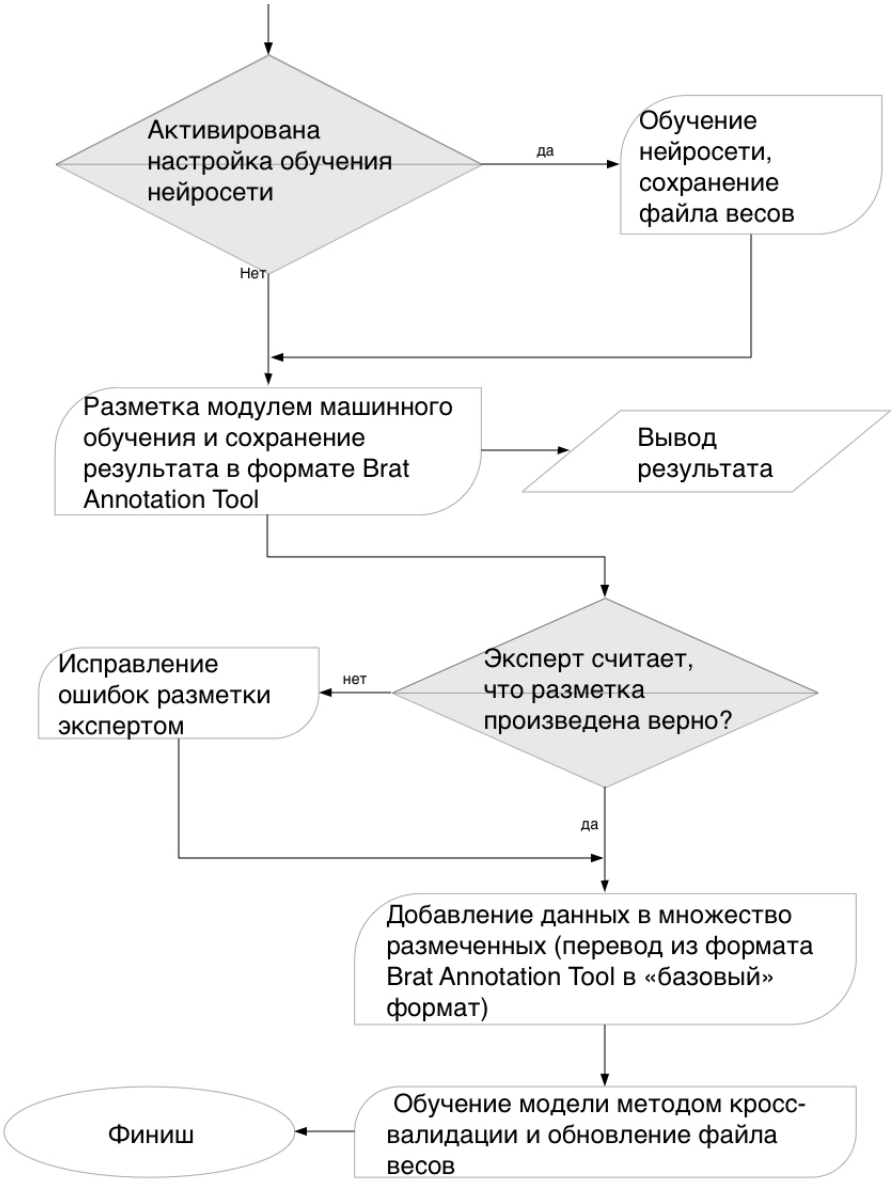



**Рис. 13:** Схема работы программного комплекса.